

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 06-243112

(43)Date of publication of application : 02.09.1994

(51)Int.Cl.

G06F 15/16

G06F 9/46

G06F 13/00

(21)Application number : 05-030613

(71)Applicant : SEIKO EPSON CORP

(22)Date of filing : 19.02.1993

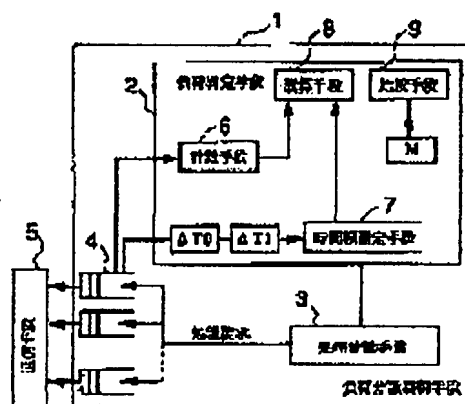
(72)Inventor : NAGASAKA FUMIO

(54) MULTIPROCESSOR DEVICE

(57)Abstract:

PURPOSE: To attain the optimum dispersion of load by detecting the load on other processors in a multiprocessor parallel processing system working in a network environment where different device types are mixed.

CONSTITUTION: A queue 4 and a load deciding means 2 are provided for each processor when the processing units are dispersed to other processors and the processing time is shortened by the parallel execution by the processors. The means 2 performs an arithmetic operation based on the length of the queue 4 and the results $\Delta T0$ and $\Delta T1$ of hitherto processing time and returns the expected value M of the end time of the next processing. A processing dispersion means 3 distributes the processing to the processor device that has the least value M and enqueues the queue 4. When either processing is complete, the means 3 records the processing time and dequeues the processing unit from the queue 4. A load dispersion means 1 consists of the means 2 and 3 and a queue 4 and knows the load of other processors with no communication and also can disperse the processing.



(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開平9-288648

(43) 公開日 平成9年(1997)11月4日

(51) Int.Cl. ⁶	識別記号	庁内整理番号	F I	技術表示箇所
G 0 6 F 15/16	3 7 0		G 0 6 F 15/16	3 7 0 N
13/00	3 5 7		13/00	3 5 7 Z

審査請求 未請求 請求項の数 7 O L (全 12 頁)

(21) 出願番号 特願平8-98174

(22) 出願日 平成8年(1996)4月19日

(71) 出願人 000005821

松下電器産業株式会社

大阪府門真市大字門真1006番地

(72) 発明者 若谷 彰良

大阪府門真市大字門真1006番地 松下電器
産業株式会社内

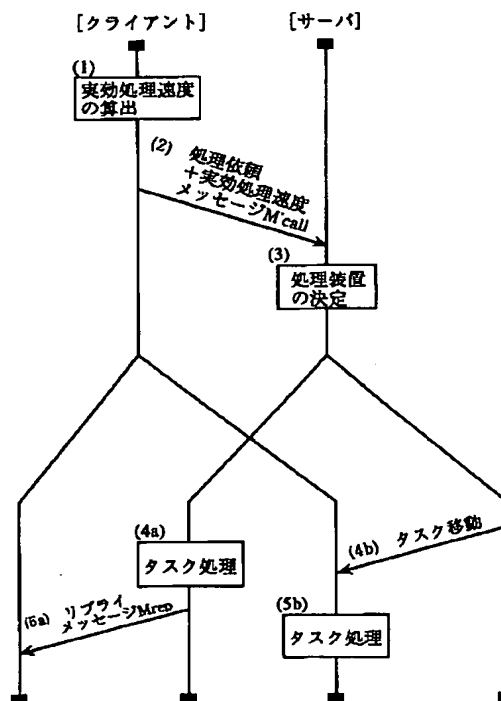
(74) 代理人 弁理士 中島 司朗

(54) 【発明の名称】 ネットワークにおける分散処理方法及びそのシステム

(57) 【要約】

【課題】 ネットワークにおける負荷状況の変動に対応した効率的な分散処理方式を提供する。

【解決手段】 クライアント1は、自らの実効処理速度を算出し(手順1)、その情報と共にサーバ11に対してサーバ11が備えるタスクの処理を依頼する(手順2)。サーバ11は、受信したクライアントと処理速度と自らの処理速度に基づいて、そのタスクに係るターンアラウンド時間を短縮するには、いずれの装置がそのタスクを処理すべきか判断する(手順3)。その判断により、サーバ自らがそのタスクを処理するか(手順4 a、5 a)、又は、そのタスクをクライアントに移動した後(手順4 b)クライアントが処理する(手順5 b)が行われる。



【特許請求の範囲】

【請求項1】 複数のタスクを並行処理し得る複数の端末装置が通信路に接続されたネットワークにおいて、一の端末装置（クライアント）が他の一の端末装置（サーバ）に備えられた特定タスクの処理結果を利用したい場合における分散処理方法であって、前記クライアントが前記結果を利用したい旨の要求を発すると、前記特定タスクに係るターンアラウンド時間を小さくするためにはサーバ又は前記クライアントのいずれがその特定タスクを処理すべきかを判断する第1ステップと、サーバが処理すべきと判断した場合には、そのサーバは前記特定タスクを処理しその結果を前記クライアントに報告し、一方、前記クライアントが処理すべきと判断した場合には、前記サーバは前記特定タスクを前記クライアントに移動しそのクライアントがその特定タスクを処理する第2ステップとからなることを特徴とする分散処理方法。

【請求項2】 前記第1ステップは、前記サーバにより行われ、現時点における前記サーバ及び前記クライアントの実効処理速度を決定する決定ステップと、決定した実効処理速度に基づいて、前記サーバが前記特定タスクを処理しその結果を前記クライアントに報告した場合に要する第1の時間と、前記サーバから前記クライアントに前記特定タスクを移動しそのクライアントがその特定タスクを処理した場合に要する第2の時間とを求めて比較することにより前記判断をする判断ステップとからなることを特徴とする請求項1記載の分散処理方法。

【請求項3】 前記決定ステップにおいては、前記サーバ及び前記クライアントの実効処理速度 R_s 、 R_c は、並行処理されているそれぞれのタスク数 N_s 、 N_c を検出し、それぞれの最大処理速度 R_{smax} 、 R_{cmax} を用いて、式 $R_s = R_{smax} / (N_s + 1)$ 及び $R_c = R_{cmax} / (N_c + 1)$ により決定し、前記判断ステップにおいては、前記第1の時間 T_1 及び第2の時間 T_2 は、前記特定タスクの処理に伴う仕事量を M_{task} 、前記通信路の通信速度を R_t 、前記報告に伴う通信量を M_{rep} 、前記タスク移動に伴う通信量を M_{mig} とすると、式 $T_1 = M_{task} / R_s + M_{rep} / R_t$ 及び $T_2 = M_{mig} / R_t + M_{task} / R_c$ により算出することを特徴とする請求項2記載の分散処理方法。

【請求項4】 前記決定ステップにおける前記クライアントの実効処理速度の決定は、そのクライアントが現時点における自らの実効処理速度を算出し、その実効処理速度を示す情報を前記要求と同時に前記サーバに通知し、その通知を受けた前記サーバにより行われることを

特徴とする請求項2記載の分散処理方法。

【請求項5】 前記検出ステップにおける前記クライアントの実効処理速度の決定は、前記要求を受けた前記サーバが前記クライアントに問い合わせ、そのクライアントが現時点における自らの実効処理速度を算出して返答し、その返答を受けた前記サーバにより行われることを特徴とする請求項2記載の分散処理方法。

【請求項6】 特定タスクを有するサーバとその特定タスクの処理結果を利用するクライアントとが通信路を介して接続された分散処理システムであって、前記クライアントは、前記サーバに前記結果を利用したい旨の要求をする手段と、現時点における自らの実効処理速度を算出して前記サーバに通知する第1算出手段と、前記サーバから前記特定タスクが移動されてきた場合には、その特定タスクを処理する手段とを備え、前記サーバは、現時点における自らの実効処理速度を算出する第2算出手段と、

前記要求を受けると、前記クライアント及び前記サーバで算出された実効処理速度に基づいて、サーバ自らが前記特定タスクを処理しその結果を前記クライアントに報告した場合に要する第1の時間と、前記特定タスクを前記クライアントに移動しそのクライアントがその特定タスクを処理した場合に要する第2の時間とを求めて比較する比較手段と、前記第1の時間が小さい場合には、サーバ自らが前記特定タスクを処理しその結果を前記クライアントに報告し、一方、前記第2の時間が小さい場合には、前記特定タスクを前記クライアントに移動する手段とを備えることを特徴とする分散処理システム。

【請求項7】 前記第1算出手段は、前記クライアントの最大処理速度 R_{cmax} を予め記憶する第1記憶手段と、現時点において並行処理しているタスク数 N_c を検出する第1検出手段と、前記第1記憶手段から読み出した最大処理速度 R_{cmax} と前記第1検出手段により検出されたタスク数 N_c とから、前記実効処理速度 R_c を、式 $R_c = R_{cmax} / (N_c + 1)$ により算出する第1速度算出手段とからなり、前記第2算出手段は、前記サーバの最大処理速度 R_{smax} を予め記憶している第2記憶手段と、現時点において並行処理しているタスク数 N_s を検出する第2検出手段と、前記記憶手段から読み出した最大処理速度 R_{smax} と前記検出手段により検出されたタスク数 N_s とから、前記実効処理速度 R_s を、式

$$R_s = R_{smax} / (N_s + 1)$$

により算出する第2速度算出手段とからなり、

前記比較手段は、

前記特定タスクの処理に伴う仕事量 M_{task} 、前記通信路の通信速度 R_t 、前記報告に伴う通信量 M_{rep} 及び前記タスク移動に伴う通信量 M_{mig} を予め記憶している第3記憶手段と、

前記要求を受けると、前記第3記憶手段から読み出した値並びに前記第1速度算出手段及び前記第2速度算出手段により算出された実効処理速度 R_c 、 R_s を用いて、前記第1の時間 T_1 及び第2の時間 T_2 を、式 $T_1 = M_{task} / R_s + M_{rep} / R_t$ 及び

$$T_2 = M_{mig} / R_t + M_{task} / R_c$$

により算出する時間算出手段と、

前記時間算出手段により算出された第1の時間 T_1 及び第2の時間 T_2 を比較する手段とからなることを特徴とする請求項6記載の分散処理システム。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明は、クライアントとサーバからなる分散処理方法及びそのシステムに関し、特に、タスク処理のターンアラウンド時間を短縮するための分散処理方式に関する。

【0002】

【従来の技術】近年、インタネット等に代表されるネットワーク通信技術の発達及びワークステーション等に代表される端末装置の高機能化に伴い、クライアント・サーバ形式による分散処理システムが一般化してきた。クライアント・サーバ形式の分散処理システムを実現する第1の従来方式として、遠隔手続き呼出しによる方式 (Remote Procedure Call、以下「RPC方式」という。) がある。例えば、“Concurrent Systems”, Addison-Wesley, 1992に記載されている。ここで、クライアントとは、ある特定のタスク (以下、「特定タスク」という。) の処理を依頼するネットワーク上の端末をいい、サーバとは、その依頼を受けたネットワーク上の端末をいうが、ここでは必ずしもサーバがその特定タスクの処理を実行するとは限らないとする。

【0003】図9(a)はRPC方式におけるクライアントとサーバとの通信のやりとりを示す説明図であり、図9(b)はそのシーケンス図である。この方式は、大きく3つの手順からなる。即ち、(1)クライアントは、特定タスクの処理を依頼するためにサーバに処理依頼メッセージを送る。(2)サーバは、依頼されたその特定タスクを実行する。(3)実行し終わると、サーバは、その結果を通知するためにクライアントにリプライメッセージを返す。

【0004】例えば、クライアントがサーバに対して、サーバが持つデータベースに検索処理およびその結果をグラフィメージで返す処理を依頼し、その生成されたイ

メージ情報をクライアントが受け取る場合が該当する。この方式の長所は、比較的低価格の端末と通信手段によってシステム全体としては高度な処理が実行される点にある。

【0005】クライアント・サーバ形式の分散処理システムを実現する第2の従来方式として、タスク移動 (マイグレーション) による方式がある。例えば、SUN microsystems社が提唱しているJAVA (“Hooked on Java”, Addison-Wesley, 1995 に記載) やNetscape社で提案されているもの等 (以下、この方式を「JAVA方式」という。) がそうである。

【0006】図10(a)はJAVA方式におけるクライアントとサーバとの通信のやりとりを示す説明図であり、図10(b)はそのシーケンス図である。この方式も、大きく3つの手順からなる。即ち、(1)クライアントは、特定タスクの処理を依頼するためにサーバに処理依頼メッセージを送る。(2)サーバは、処理手続きと必要データをパッケージした特定タスクの移動を行う。即ち、特定タスク本体をメッセージとして構築し、リプライメッセージとしてクライアントに返信する。

(3)その特定タスクを受け取ったクライアントは、自らその特定タスクを実行する。

【0007】この方式の長所は、クライアントにタスク処理を行わせることにより、処理が集中しがちなサーバでの負荷を軽減させることができ、これによってシステム全体における負荷分散が図れるという点にある。

【0008】

【発明が解決しようとする課題】しかしながら、上記の従来方式では、クライアントやサーバでの負荷の変動に伴い、必ずしも、ターンアラウンド時間、即ち、処理の依頼が発生してからその処理結果を利用できる状態になるまでの時間が最小になるとは限らないという問題点がある。

【0009】図11は、その問題点を説明するためのシステム構成を示すモデル図である。本図に示されるように、いま、システムの処理性能やデータ転送速度、負荷状況をモデル化することにより、以下の通り、各分散処理方式におけるターンアラウンド時間を評価する。まず、各端末装置ごとに、現在処理されているタスクに新たに1個のタスクが加わった場合に、各タスクが処理される速度、即ち、実効処理速度を定義する。

【0010】クライアントの処理速度の最大値を R_{cmax} (仕事単位/秒)、クライアントで並行処理しているタスク数を N_c (個) とすると、クライアントの実効処理速度 R_c (仕事単位/秒・タスク) は、

$$R_c = R_{cmax} / (N_c + 1) \quad \dots \quad (式1)$$

で表されるものとし、同様に、サーバの処理速度の最大値を R_{smax} (仕事単位/秒)、サーバで並行処理しているタスク数を N_s (個) とすると、サーバの実効処理速度 R_s (仕事単位/秒・タスク) は、

$$Rs = R_{smax} / (Ns + 1) \quad \dots \quad (式2)$$

で表されるものとする。

【0011】ここで、最大処理速度 R_{cmax} 、 R_{smax} とは、各端末装置固有の処理能力を示す単位であり、各装置が有するCPUのSPECmark等に相当し、仕事単位とは、タスクの処理サイズを表す単位であり、実効処理速度が1仕事単位/秒である装置が1秒間に処理できる正規化された演算量に相当する。さらに、特定タスクの仕

$$T_{rpc} = M_{call} / R_t + M_{task} / R_s + M_{rep} / R_t \quad \dots \quad (式3)$$

と表され、一方、J A V A方式におけるターンアラウンド時間 T_{java} (秒) も、上述の3つの手順に対応する時

$$T_{java} = M_{call} / R_t + M_{mig} / R_t + M_{task} / R_c \quad \dots \quad (式4)$$

と表される。

【0013】具体的な数値を用いて両方式でのターンアラウンド時間を評価すると、以下の通りである。今、システムの性能として、クライアントの処理速度の最大値 R_{cmax} が40、サーバの処理速度の最大値 R_{smax} が100、通信路の通信速度 R_t が5であり、各通信量・仕事量として、特定タスクの仕事量 M_{task} が200、依頼メッセージの通信量 M_{call} が5、リプライメッセージの通信量 M_{rep} が5、タスク移動に伴う特定タスクの通信量 M_{mig} が100であるとする。

【0014】そして、第1の負荷状況として、クライアントで並行処理しているタスク数 N_c が1、サーバで並行処理しているタスク数 N_s が19の場合を考える。式1及び式2より、クライアント及びサーバの実効処理速度 R_c 及び R_s は、

$$R_c = 40 / (1 + 1) = 20$$

$$R_s = 100 / (19 + 1) = 5$$

となるので、式3及び式4より、RPC方式及びJ A V A方式におけるターンアラウンド時間 T_{rpc} 及び T_{java} は、

$$T_{rpc} = 5 / 5 + 200 / 5 + 5 / 5 = 42 \text{ (秒)}$$

$$T_{java} = 5 / 5 + 100 / 5 + 200 / 20 = 31 \text{ (秒)}$$

となる。図12は、この結果をタイムチャートで示したものである。

【0015】以上より、この第1の負荷状況においては、J A V A方式による場合の方がRPC方式に比べてターンアラウンド時間が小さくなるので、J A V A方式がより好ましい分散処理方式と言える。続いて、サーバで並行処理しているタスク数 N_s がそれまでの19から9に減少した第2の負荷状況を考える。

【0016】第1の負荷状況の場合と同様に算出すると、クライアントの実効処理速度 R_c は20のまま変動しないが、サーバの実効処理速度 R_s は、

$$R_s = 100 / (9 + 1) = 10$$

と増加するので、RPC方式におけるターンアラウンド時間 T_{rpc} は、

$$T_{rpc} = 5 / 5 + 200 / 10 + 5 / 5 = 22 \text{ (秒)}$$

事量を M_{task} (仕事単位)、通信路の通信速度を R_t (通信単位/秒)、依頼メッセージの通信量を M_{call} (通信単位)、リプライメッセージの通信量を M_{rep} (通信単位)、タスク移動に伴う特定タスクの通信量を M_{mig} (通信単位) と定義する。

【0012】すると、RPC方式におけるターンアラウンド時間 T_{rpc} (秒) は、上述の3つの手順に対応する時間の合計になるので、

$$T_{rpc} = M_{call} / R_t + M_{task} / R_s + M_{rep} / R_t \quad \dots \quad (式3)$$

間の合計になるので、

と減少する。一方、J A V A方式におけるターンアラウンド時間 T_{java} は31秒のまま変動しない。図13は、この結果をタイムチャートで示したものである。

【0017】以上より、この第2の負荷状況においては、第1の負荷状況の場合と異なり、RPC方式による場合の方がJ A V A方式に比べてターンアラウンド時間が小さくなるので、RPC方式がより好ましい分散処理方式と言える。このように、システムの負荷状況の変動に伴い、両方式におけるターンアラウンド時間の大小関係が反転するが、従来のいずれの方式であっても、負荷の変動には依存しない固定的な方式であるために、必ずしも他方の方式よりも小さいターンアラウンド時間が提供されるとは言えない。即ち、負荷の状況によっては、急激にそのパフォーマンスが落ちるという問題点がある。

【0018】そこで、本発明は、上記問題点を解決するためになされたものであり、システムの負荷状況が変動する場合であっても最小のターンアラウンド時間が得られる効率的な分散処理方法及びそのシステムを提供することを目的とする。

【0019】

【課題を解決するための手段】上記目的を達成するために、本発明に係る分散処理方法及びシステムによれば、クライアントがサーバに備えられた特定タスクの処理結果を利用したい旨の要求を発すると、まず、前記特定タスクに係るターンアラウンド時間を小さくするためにはサーバ又はクライアントのいずれがその特定タスクを処理すべきかが判断され、次に、サーバが処理すべきと判断された場合にはその特定タスクはサーバによって処理された後にその結果が前記クライアントに報告され、一方、前記クライアントが処理すべきと判断された場合にはその特定タスクはサーバからクライアントに移動された後にその特定タスクはクライアントによって処理される。

【0020】これにより、クライアントからの処理要求が発生する度に、その処理に要するターンアラウンド時間をより小さくするための処理方式が採用されるので、サーバ又はクライアントのいずれかによって固定的に処

理されていた従来方式に比べ、そのターンアラウンド時間が短縮され、システムの負荷状況の変動に対応した効率的な分散処理システムが実現される。

【0021】

【発明の実施の形態】以下、本発明の実施の形態について図面を用いて詳細に説明する。

（実施の形態1）図1は、実施の形態1に係る分散処理システムのハードウェア構成を示すブロック図であり、ネットワークシステム全体の中の関連する2台の端末装置1、11のみを示している。

【0022】本システムは、通信路20を介して接続されたクライアント1とサーバ11から構成される。クライアント1及びサーバ11は、それぞれプロセッサ2、12、メモリ3、13及び通信部4、14から構成される。プロセッサ2、12は、CPU等であり、装置全体を制御したり複数のタスクを並列処理したりする。

【0023】メモリ3、13は、ROM、RAM及びハードディスク等からなり、後述する制御プログラム、システム性能パラメータ、タスク等を予め保持し、また、処理されるタスクの作業域としても用いられる。通信部4、14は、イーサネット用のインタフェースカード等であり、通信路20と装置1、11との中継をする。

【0024】なお、クライアント1及びサーバ11は、汎用のワークステーション等の端末装置のハードウェア構成と同じであるが、メモリ3、13に本実施形態を実現する固有の制御プログラム等を保持している点で汎用の端末装置と異なる。以上のように構成された分散処理システムにおける動作について説明する。ここで、従来技術での説明と同様の状況を考える。即ち、クライアント1は、サーバ11のメモリ13に保持されている特定タスクの処理結果を得たいとする。この場合における分散処理方式を説明する。

【0025】図2は、本実施形態の分散処理方式におけるクライアント1とサーバ11との通信のやりとりを示すシーケンス図であり、従来の図9（b）及び図10（b）に対応する。このシーケンス図は、2つの異なる場合における流れを同時に示している。即ち、以下の手順1～3、4a、5aからなる場合と、手順1～3、4b、5bからなる2つの場合である。

（手順1）まず、クライアント1は、特定タスクの処理の依頼に先立ち、自らの実効処理速度を算出する。

【0026】具体的には、プロセッサ2は、自らの処理負荷、即ち、メモリ3にロードされ並列処理されているタスクの数 N_c を検出し、さらに、メモリ3に格納されている最大処理速度 R_{cmax} を読み出し、式1に従って、実効処理速度 R_c を算出する。

（手順2）続いて、クライアント1は、算出した実効処理速度 R_c とサーバ11が保持する特定タスクの処理結果を利用したい旨とを一つのメッセージM'callにしてサーバ11に送信する。

【0027】具体的には、プロセッサ2は、送信元、即ち、クライアント1のネットワークアドレス、サーバが有する特定タスクを識別する情報等を通信部4を介してサーバ11に送信する。

（手順3）メッセージM'callを受信したサーバ11は、その特定タスクに係るターンアラウンド時間を短縮するには、その特定タスクを自ら処理すべきかクライアント1が処理すべきかを判断する。

【0028】具体的には、プロセッサ12は、通信部14を介して受信したメッセージM'callにより特定された特定タスクが保持されたメモリ13を参照することにより、この特定タスクの仕事量Mtask、タスク移動に伴う通信量Mmig及びリプライメッセージの通信量Mrepを決定すると共に、そのメッセージM'callに含まれるクライアント1の実効処理速度 R_c を抽出する。

【0029】続いて、プロセッサ12は、通信部14による定期的な通信路20の観察に基づいて、現在における通信路20の平均通信速度 R_t を検出する。そして、プロセッサ12は、クライアント1の場合と同様にし、自らの処理負荷、即ち、実効処理速度 R_s を算出する。以上の値に基づいて、プロセッサ12は、式3及び式4のそれぞれの第2項及び第3項からなる以下の2種類の時間 T_1 、 T_2 を算出し比較する。

【0030】

$$T_1 = M_{task} / R_s + M_{rep} / R_t \quad \dots \quad (式5)$$

$$T_2 = M_{mig} / R_t + M_{task} / R_c \quad \dots \quad (式6)$$

なお、式5で表される T_1 と式6で表される T_2 との大小関係は、式3で表されるRPC方式によるターンアラウンド時間 T_{rpc} と式4で表されるJAVA方式によるターンアラウンド時間 T_{java} との大小関係と同じになる。なぜなら、式3と式4のそれぞれの第1項は共通だからである。

【0031】比較の結果、 $T_1 < T_2$ の場合には、サーバ11は、サーバ自らこの特定タスクを処理すべきと判断し、以下の手順4a、5aを実行する。

（手順4a）サーバ11は、特定タスクを処理する。具体的には、プロセッサ12は、クライアント1から依頼された特定タスクをメモリ13の保持部から作業用領域に移し、既に実行しているタスクに加えてタイムシェアリングにより実行する。

（手順5a）特定タスクの処理を終えると、サーバ11は、その処理によって得られた結果をリプライメッセージとしてクライアント1に返信する。

【0032】具体的には、プロセッサ12は、クライアント1からの要求に対する返答として、その特定タスクの実行により得られたデータを通信部14を介してクライアント1に送信する。以上の手順1～3、4a、5aにより、クライアント1から依頼された特定タスクはサーバ11によって処理されるので、結果的に従来のRPC方式と同様の分散処理方式が行われたことになる。

【0033】一方、手順3での比較において、 $T1 \geq T2$ の場合には、サーバ11は、サーバ11自らではなくクライアント1が特定タスクを処理すべきと判断し、上記手順4a、5aに代えて以下の手順4b、5bを実行する。

(手順4b)サーバ11は、特定タスクをクライアント1に移動する。

【0034】具体的には、プロセッサ12は、クライアント1から依頼された特定タスクをメモリ13の保持部から読み出し、通信部14を介してクライアント1に転送する。なお、ここで転送される情報には、特定タスク自体、即ち、処理手続き及び必要な関連データだけでなく、手順2によるクライアント1からの要求に対する返答としてタスク移動が行われた旨のメッセージ等も含まれる。

(手順5b)その特定タスクを受信したクライアント1は、自らその特定タスクを処理する。

【0035】具体的には、プロセッサ2は、通信部4を介してメモリ3の作業領域にロードした特定タスクを、既に実行しているタスクに加えてタイムシェアリングにより実行する。以上の手順1～3、4b、5bにより、クライアント1から依頼された特定タスクはクライアント1自らによって処理されるので、結果的に従来のJ A V A方式と同様の分散処理方式が行われたことになる。

【0036】このように、本方式によれば、システムにおける処理負荷の状況、即ち、クライアント1及びサーバ11において並列処理されているタスク数を考慮することにより、 $T1 < T2$ 、即ち、 $Trpc < Tjava$ と判断された場合には、従来のR P C方式によるタスク処理が行われ、一方、 $T1 \geq T2$ 、即ち、 $Trpc \geq Tjava$ と判断された場合には、従来のJ A V A方式によるタスク処理が行われる。

【0037】これによって、システムの処理負荷の状況に応じ、R P C方式又はJ A V A方式のうち、より小さいターンアラウンド時間が得られる方式が動的に決定されたことになる。図3は、以上のシーケンスにおけるサーバ11のみの動作を示すフローチャートである。

【0038】サーバ11は、クライアント1から送られてきた処理要求メッセージM'callを受信した後(ステップS31)、通信部14で通信路20の通信速度Rtを検出する(ステップS32)。そして、受信したメッセージM'callから、実行すべき特定タスクの内容を決定し、クライアント1の実効処理速度Rcを抽出する(ステップS33)。

【0039】さらに、サーバ11の実効処理速度Rsを求め(ステップS34)、タスク移動に伴うタスクの通信量Mmig、処理に必要な演算数Mtaskおよびタスク処理の終了時にリプライに必要となるメッセージの通信量Mrepを求める(ステップS35)。以上の情報を用い、式5で表されるT1と式6で表されるT2とを比較

し(ステップS36)、前者が小さい場合は、サーバ11内でタスク処理を行なった後に(ステップS37)その結果をクライアント1にリプライし(ステップS38)、そうでない場合は、クライアント1にタスク本体を移動してクライアント1内でその特定タスクの処理を行なうようにする(ステップS39)。

【0040】図4は、図2に示されたシーケンスにおけるクライアント1のみの動作を示すフローチャートである。まず、クライアント1は、実効処理速度Rcを求め(ステップS41)、それを処理依頼メッセージM'callに付加してサーバ11に送信する(ステップS42)。

【0041】続いて、サーバ11からのリプライを受信し(ステップS43)、それが特定タスク本体であれば(ステップS44)、クライアント1内でその特定タスクを実行し(ステップS45)、そうでなければ、そのリプライを特定タスクの処理結果として利用する。以上のシーケンスについて、従来技術での説明と同様の具体的な数値を用いてその流れを説明する。

【0042】即ち、システムの性能として、 $Rcmax = 40$ 、 $Rsmax = 100$ 、 $Rt = 5$ であり、通信量・仕事量として、 $Mtask = 200$ 、 $M'call = 5$ 、 $Mrep = 5$ 、 $Mmig = 100$ であり、第1の負荷状況として、 $Nc = 1$ 、 $Ns = 19$ の場合を考える。すると、
 $Rc = 40 / (1 + 1) = 20$
 $Rs = 100 / (19 + 1) = 5$
 より、

$$T1 = 200 / 5 + 5 / 5 = 41$$

$$T2 = 100 / 5 + 200 / 20 = 30$$

となるので、上記手順3においては、 $T1 \geq T2$ と判断され、クライアント1が特定タスクを処理すべきと判断されるので、その後、上記手順4b、5b、即ち、従来のJ A V A方式が採られる。

【0043】図5は、この第1の負荷状況における本方式のタイムチャートを、従来の固定的なR P C方式との比較において示す図である。式3及び式4より、本方式により得られるターンアラウンド時間(31秒)は、従来方式の場合(42秒)よりも小さい。続いて、 $Ns = 9$ に減少した第2の負荷状況を考える。

$$【0044】Rs = 100 / (9 + 1) = 10$$

に増加するので、

$$T1 = 200 / 10 + 5 / 5 = 21$$

と減少するが、T2はそのまま(30)であるので、上記手順3においては、 $T1 < T2$ と判断され、クライアント1が特定タスクを処理すべきと判断されるので、その後、上記手順4a、5a、即ち、従来のR P C方式が採られる。

【0045】図6は、この第2の負荷状況における本方式のタイムチャートを、従来の固定的なJ A V A方式との比較において示す図である。式3及び式4より、本方

式により得られるターンアラウンド時間(22秒)は、従来方式の場合(31秒)よりも小さい。以上のように、本方式によれば、タスク処理の要求が発生した時点におけるシステムでの処理負荷の状況が考慮されるので、RPC方式又はJAVA方式のうち、より小さいターンアラウンド時間が得られる分散処理方式が動的に決定される。

(実施の形態2)次に、本発明の実施の形態2に係る分散処理システムを説明する。

【0046】本実施形態は、クライアントからの処理依頼が発生してからその処理をすべき装置を決定するまでの手順が実施形態1と異なり、他の点においては実施形態1と同じである。従って、ここでは、実施形態1と異なる点のみ説明する。図7は、本実施形態の分散処理方式におけるクライアント1とサーバ11との通信のやりとりを示すシーケンス図である。実施形態1における図2と比較して判るように、本実施形態では、実施形態1の手順1、2に代えて異なる手順1~4を採用し、それ以降の手順は同一である。

(手順1)まず、クライアント1は、サーバ11に対して、サーバ11が保持する特定タスクの処理結果を利用したい旨のメッセージのみを送信する。ここでは、実施形態1と相違し、このメッセージにはクライアント1の実効処理速度Rcの情報は含まれない。

(手順2)そのメッセージを受信したサーバ11は、自らの処理負荷が軽くなった頃等を見計らって、クライアント1に実効処理速度Rcを問い合わせる。

(手順3)問い合わせを受けたクライアント1は、実施形態1の場合と同様の方法により現時点での自らの実効処理速度Rcを算出する。

(手順4)そして、クライアント1は、その実効処理速度Rcをサーバ11に送信する。

(手順5、6a、7a、6b、7b)これらのクライアント1及びサーバ11での動作は、それぞれ実施形態1の手順3、4a、5a、4b、5bと同一である。

【0047】なお、図8は、以上のシーケンスにおけるサーバ11のみの動作を示すフローチャートであり、実施形態1における図3に対応する。以上のように、本実施形態は、実施形態1と比較し、手順5におけるサーバ11での判断のために必要とされる情報の準備方法(手順1~4)が異なるだけで、手順5における判断内容やその結果による以降の流れは同一であるので、実施形態1と同様に、システムでの処理負荷の状況を考慮した最適な分散処理方式が動的に行われるのは言うまでもない。

【0048】また、クライアント1は、処理依頼メッセージを発行する毎に実効処理速度Rcを計算するのではなくサーバから問い合わせられた時のみ計算するので、クライアント1の負荷は小さくなる。さらに、サーバ11は、自分にとって都合のよいタイミングでクライアン

ト1の実効処理速度Rcを獲得できるので、手順6a又は7bにおいて実際に特定タスクが処理される時刻に近い時刻において手順5での判断を行うことが可能となり、刻々と変化する負荷状況にあっても、より精度の高い処理方式が決定される。

【0049】以上、本発明に係る分散処理システムについて、実施形態に基づいて説明したが、本発明はこれら実施形態に限られないことは勿論である。即ち、

(1)上記実施形態においては、プロセッサの処理能力が処理対象となる各タスクに均等に分配されることを前提に、各端末装置の実効処理速度は式1及び式2で表されるとしたが、この実効処理速度は、これらの式で表されるものに限定されない。例えば、優先順位を伴う重み付けされたスケジューリングが行われる端末装置であれば、その重み付けを考慮した式となる。

(2)本分散処理方式における処理装置の決定は、サーバ11によって行われたが、この装置に限定されるものではなく、例えば、クライアント1や第3の端末装置によって行われるとするシステムとすることも考えられる。

【0050】

【発明の効果】以上の説明から明らかなように、本発明に係る分散処理方法及びシステムによれば、クライアントがサーバに備えられた特定タスクの処理結果を利用したい旨の要求を発すると、先ず、前記特定タスクに係るターンアラウンド時間を小さくするためにはサーバ又はクライアントのいずれがその特定タスクを処理すべきかが判断され(第1ステップ)、次に、サーバが処理すべきと判断された場合にはその特定タスクはサーバによって処理された後にその結果が前記クライアントに報告され、一方、前記クライアントが処理すべきと判断された場合にはその特定タスクはサーバからクライアントに移動された後にその特定タスクはクライアントによって処理される(第2ステップ)。

【0051】これにより、クライアントからの処理要求が発生する度に、その処理に要するターンアラウンド時間をより小さくするための処理方式が行われるので、サーバ又はクライアントのいずれかによって固定的に処理されていた従来方式に比べ、システムの負荷状況の変動に対応した効率的な分散処理方法及びシステムが実現されるという効果がある。

【0052】ここで、サーバ又はクライアントのいずれが処理すべきかを決定する方法として、先ず、その時点におけるサーバ及びクライアントの実効処理速度を検出し、次に、その実効処理速度に基づいて、サーバが特定タスクを処理しその結果を前記クライアントに報告した場合に要する第1の時間と、サーバからクライアントに特定タスクを移動しそのクライアントがその特定タスクを処理した場合に要する第2の時間を求めて比較することにより行うことができる。

【0053】また、サーバ及びクライアントの実効処理速度 R_s 、 R_c は、並行処理されているそれぞれのタスク数 N_s 、 N_c を検出し、それぞれの最大処理速度 R_{smax} 、 R_{cmax} を用いて、式 $R_s = R_{smax} / (N_s + 1)$ 及び $R_c = R_{cmax} / (N_c + 1)$

により決定し、第1の時間 T_1 及び第2の時間 T_2 は、特定タスクの処理に伴う仕事量を M_{task} 、通信路の通信速度を R_t 、報告に伴う通信量を M_{rep} 、タスク移動に伴う通信量を M_{mig} とすると、式 $T_1 = M_{task} / R_s + M_{rep} / R_t$ 及び

$T_2 = M_{mig} / R_t + M_{task} / R_c$

により算出することもできる。

【0054】これにより、刻々と変化する両装置での処理能力を考慮した柔軟で精度の高い分散処理方式が可能となる。また、前記第1ステップは、クライアントが現時点における自らの実効処理速度を算出し、その実効処理速度を示す情報を前記要求と同時にサーバに通知し、サーバが前記判断を行うとすることもできる。

【0055】これによって、クライアントからの処理要求と共にクライアントの処理能力がサーバに同時に通知されるので、この分散処理のための通信トラフィックの増加を最小限に抑えることができる。また、前記第1ステップは、前記要求を受けたサーバがクライアントに問い合わせ、クライアントが現時点における自らの実効処理速度を算出して返答し、サーバが前記判断を行うとすることもできる。

【0056】これによって、サーバは、自らの都合のよいタイミングでクライアントの処理能力を知ることができるので、処理すべき装置をより的確に決定することができるという効果がある。

【図面の簡単な説明】

【図1】本発明の実施の形態1及び2に係る分散処理システムのハードウェア構成を示すブロック図である。

【図2】実施形態1の分散処理方式におけるクライアント1とサーバ11との通信のやりとりを示すシーケンス図である。

【図3】図2のシーケンスにおけるサーバ11のみの動作を示すフローチャートである。

【図4】図2のシーケンスにおけるクライアント1のみの動作を示すフローチャートである。

【図5】実施形態1の第1の負荷状況におけるタイムチャートを、従来の固定的なRPC方式との比較において示す図である。

【図6】実施形態1の第2の負荷状況におけるタイムチャートを、従来の固定的なJAVA方式との比較において示す図である。

【図7】実施形態2の分散処理方式におけるクライアント1とサーバ11との通信のやりとりを示すシーケンス図である。

【図8】図7のシーケンスにおけるサーバ11のみの動作を示すフローチャートである。

【図9】図9(a)は従来のRPC方式におけるクライアントとサーバとの通信のやりとりを示す説明図であり、図9(b)はそのシーケンス図である。

【図10】図10(a)は従来のJAVA方式におけるクライアントとサーバとの通信のやりとりを示す説明図であり、図10(b)はそのシーケンス図である。

【図11】各分散処理方式におけるターンアラウンド時間を評価するためのシステム構成のモデル図である。

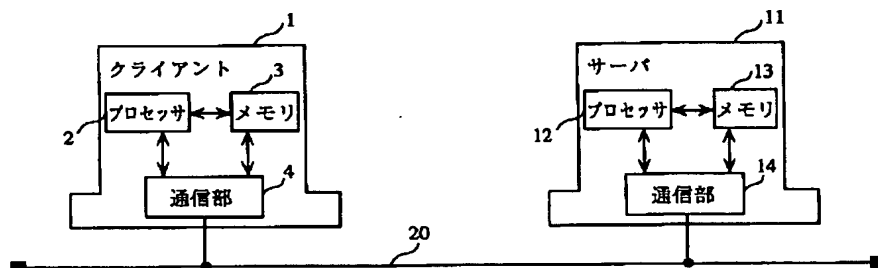
【図12】JAVA方式による場合の方がRPC方式に比べてターンアラウンド時間が小さくなる場合の従来例のタイムチャートである。

【図13】RPC方式による場合の方がJAVA方式に比べてターンアラウンド時間が小さくなる場合の従来例のタイムチャートである。

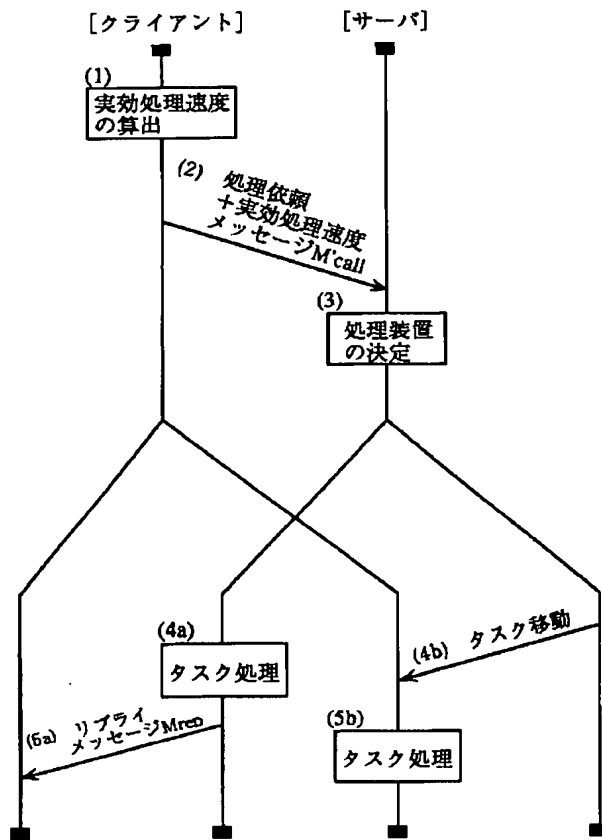
【符号の説明】

- 1 クライアント
- 2 プロセッサ
- 3 メモリ
- 4 通信部
- 11 サーバ
- 12 プロセッサ
- 13 メモリ
- 14 通信部
- 20 通信路

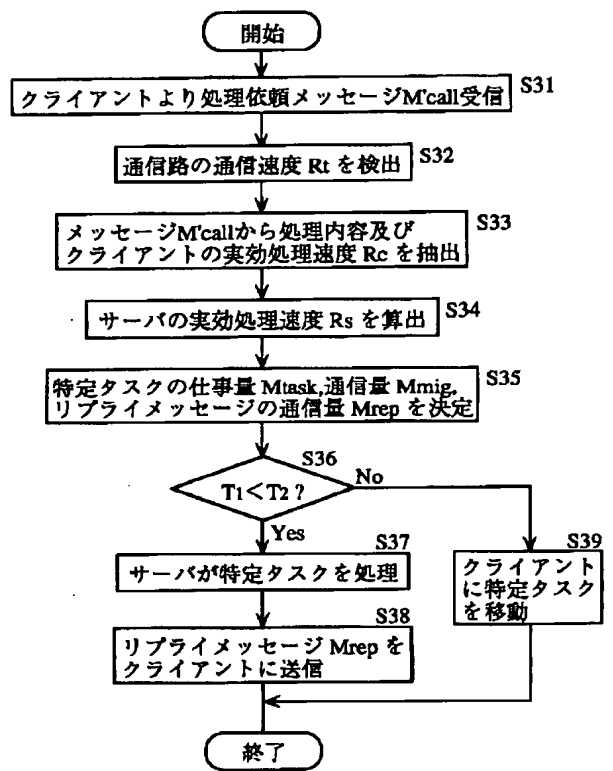
【図1】



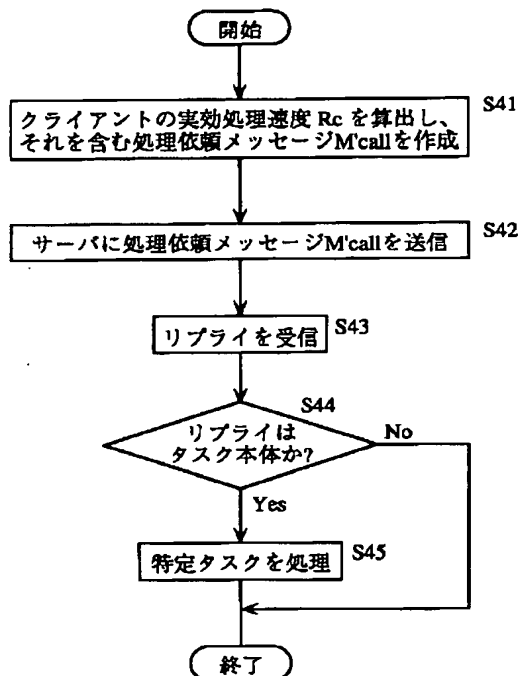
【図2】



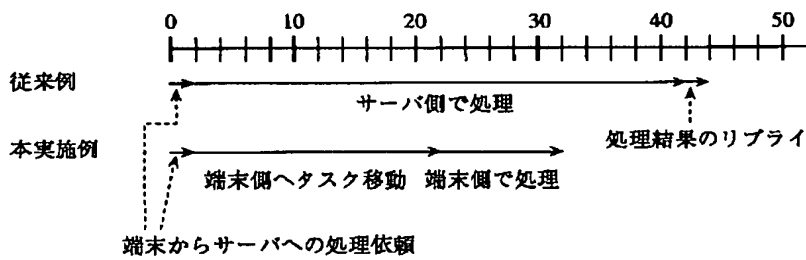
【図3】



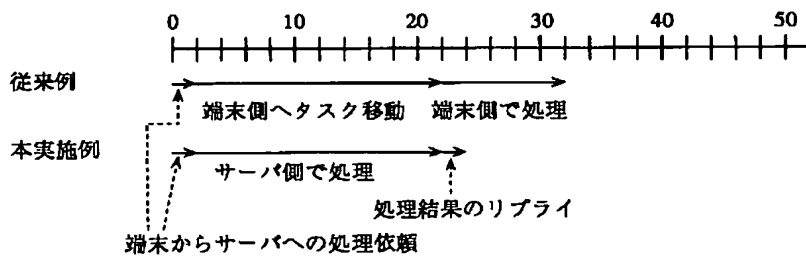
【図4】



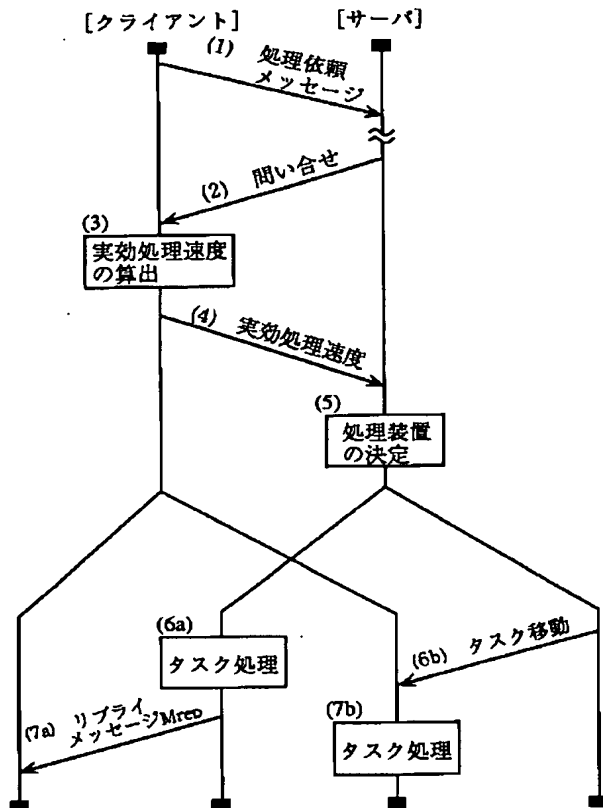
【図5】



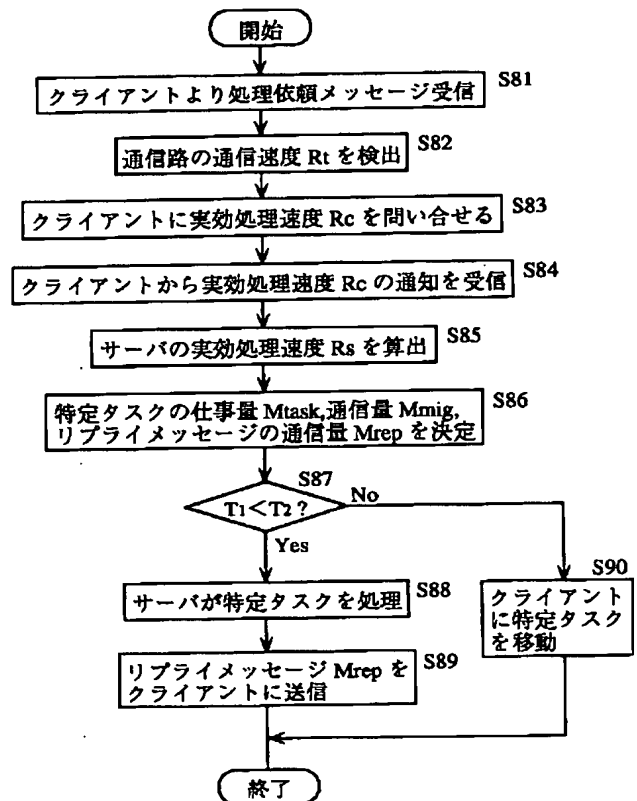
【図6】



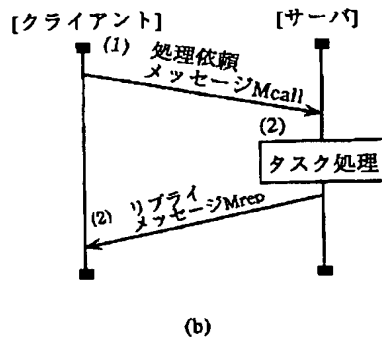
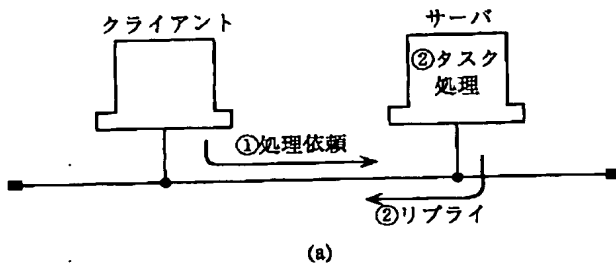
【図7】



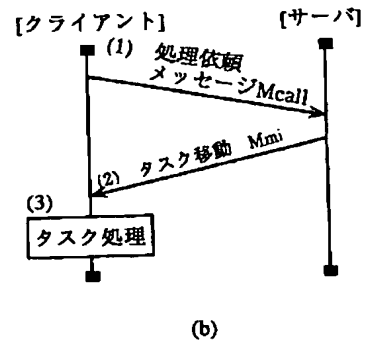
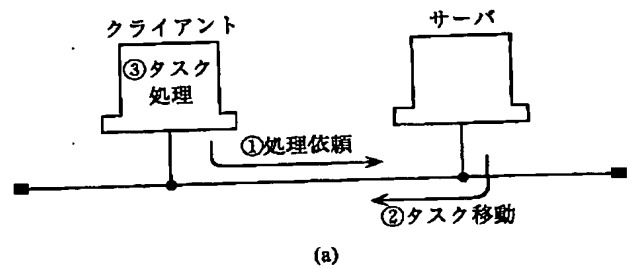
【図8】



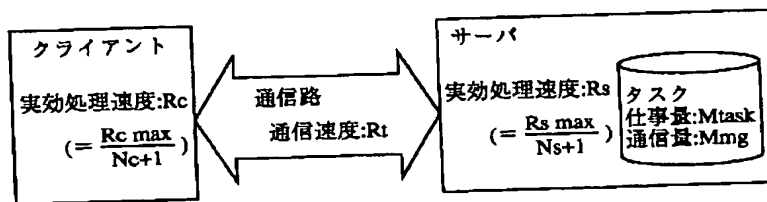
【図9】



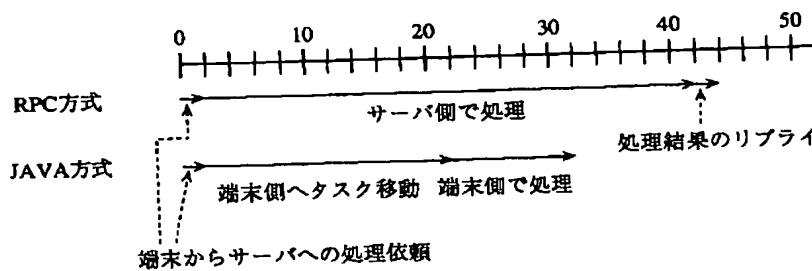
【図10】



【図11】



【図12】



【図13】

